(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number : **0 609 975 A2**

(12) EUROPEAN PATENT APPLICATION

(21) Application number : **94300017.4**

(51) Int. Cl.[5] : **G06K 15/00**

(22) Date of filing : **04.01.94**

(30) Priority : **04.01.93 US 74**

(43) Date of publication of application :
10.08.94 Bulletin 94/32

(84) Designated Contracting States :
DE FR GB

(71) Applicant : **XEROX CORPORATION**
Xerox Square
Rochester New York 14644 (US)

(72) Inventor : **Catapano, David A.**
339 Hazelwood Terrace
Rochester, New York 14609 (US)

Inventor : **Reilly, Paul E.**
460 Collinwood Court
Santa Clara, California 95054 (US)
Inventor : **Zell, Thomas B.**
8048 Sky View Path
Victor, New York 14564 (US)
Inventor : **Hsu, Lillian-Liu**
135 Aspen Drive
Rochester, New York 14625 (US)
Inventor : **Baxter, Eric W.**
32 Delaware Street
Rochester, New York 14607 (US)
Inventor : **Simpson, Mark F.**
22451 Franklin Court
Mountain View, California 94040 (US)

(74) Representative : **Hill, Cecilia Ann et al**
Rank Xerox Patent Department,
Albion House,
55-59 New Oxford Street
London WC1A 1BS (GB)

(54) Apparatus and method for processing a stream of image data in a printing system.

(57) A printing system for producing prints from a job represented by a stream of image data written in a page description language and having a token (256') expressed as a plurality of bits. The stream of image data is generated by and transmitted from an image data source (40,42). The printing system comprises an input section (25), communicating with the image data source, for receiving the stream of image data. The input section includes an arrangement (DIF 222) for reading a block of image data to determine (227,228) the presence of the token in the block of image data. An area (72,82), communicating with the input section, is adapted to store the stream of image data. An arrangement, responsive to the reading means determining the presence of the token, is adapted to initiate a selected operation (262',266',270') in the printing system after the input section receives a portion of the stream of image data. A parser (60), communicating with the storage area, is adapted to parse the stream of image data to separate the stream into a plurality of image-related components.

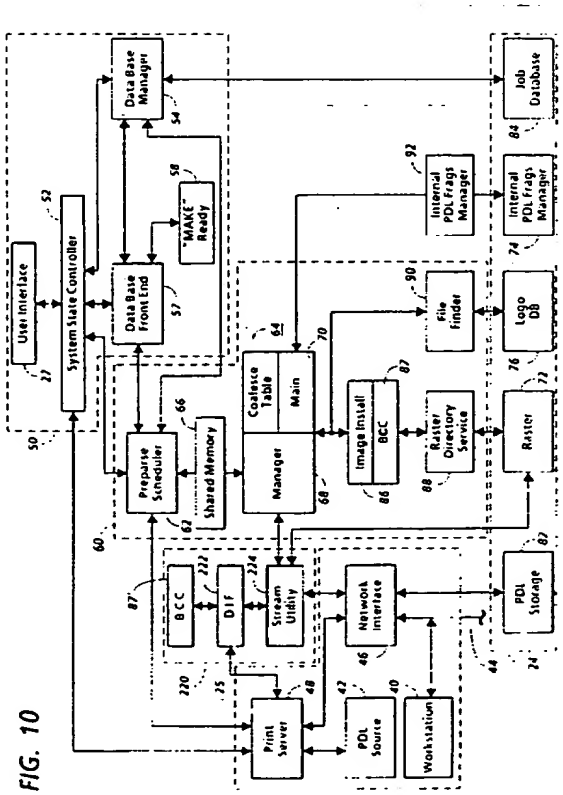


FIG. 10

EP 0 609 975 A2

The present invention relates generally to a technique for printing a stream of image data written in a printer page description language and more specifically to an image filtering arrangement which selectively removes one or more images from the stream of image data to thereby form a modified stream of image data.

Network printing systems are becoming prevalent in office settings where extensive document processing is performed. In one example of such network printing systems, a client at an input, such as a workstation, sends electronic documents that comprise a job over a local area network (LAN) to one of the printers selected for printing of the job. In particular, LANs provide a means by which users running dedicated processors are able to share resources such as printers, file servers and scanners. Integration of shared resources has been a problem addressed by LAN managers. LAN managers have made different network protocols transparent to devices running different network protocols. LANs also have a variety of print drivers emitting different page description languages (PDLs), which are directed to specific print devices.

In an exemplary network printing system, image data is transmitted to an electronic printing system, such as a network DocuTech® electronic printer manufactured by Xerox® Corporation, in the form of a stream of image data expressed in terms of the PDL. The PDL stream can include, among other things, an image, such as a bitmap, or a reference to one or more images existing outside of the PDL stream. Preferably, suitable software, is employed to "take apart" an input document so that the PDL is "parsed" into various image related components with a preparser, as discussed in EP-A-0,574,224 (D/92352).

As discussed in EP-A-0,574,224, the decomposer executes the PDL to generate imaging primitives. In doing this, the decomposer, for example uses an arrangement for parsing the PDL into various image related components. The types of operations required to generate imaging primitives include, among others, binding fonts with requested fonts, any image processing on pictorial information, and/or converting line art/graphics (including bitmaps) to lower level primitives.

The decomposer of the exemplary network printing system may be designed to receive only one job file at a time. While the system can store a plurality of job files, in anticipation of parsing them, such storage can result in less than desirable machine output, or, more specifically, in reduced printing speeds. That is, the amount of image data contained in a given job file can be relatively large, if not immense, so that storing the job file out, rather than transmitting it directly from the network input to the parsing arrangement, can result in loss of time since, due to the amount of image data in the job file, a significant delay inevitably occurs when the job file is retrieved from storage for decomposing.

Additionally, due to the size of various images in the job file, and the finite resources of the decomposer, a significant amount of time can be expended in parsing the images of the PDL stream. Delays generated by relatively slow parsing can be compounded in a system where multiple jobs are queued up for decomposing. For example, in a situation where n jobs are queued up for decomposing, the time that it takes for the decomposer to finally get to the nth. job will be affected by the amount of time that it takes to parse each of the n-1 job files which is ahead of the nth job file in the queue.

It would be desirable to provide an intelligent spooling arrangement to enable throughput of the printing system to be maximized.

In accordance with the present invention there is provided a printing system for producing prints from a job represented by a stream of image data written in a page description language and having a token expressed as a plurality of bits, the stream of image data being generated and transmitted from an image data source. The printing system comprises: an input section, communicating with the image data source, for receiving the stream of image data, the input section including means for reading a block of image data to determine the presence of the token in the block of image data; means, communicating with the input section, for storing the stream of image data; means, responsive to the reading means determining the presence of the token, for initiating a selected operation in the printing system after the input section receives a portion of the stream of image data; and a parser, communicating with the storage section, for parsing the stream of image data to separate the stream into a plurality of image-related components.

The storing means of a system in accordance with the present invention may include disk memory and may also include cache memory.

The input section may comprise an image filtering device; and means for controlling the flow of image data to and from said filtering device.

The image data source may comprise a network across which the first stream of image data is communicated, wherein said input section includes means for coupling said input section with the network. Said coupling means may be adapted to permit parallel multi-client access to said input section.

The said initiating means may comprise means for setting up a device having a job characteristic which defines the job for printing, said initiating means being in communication with the device. The device may be disposed remotely of said printing system. The job characteristic of the device may be a font type or a finishing attribute.

In accordance with the present invention, there is also provided a method of producing prints with a printing system from a job represented by a stream of page description language and having a token expressed as a plurality of bits, the stream of image data being generated with an image data source and transmitted from the image data source, comprising the steps of reading a block of image data to determine whether the token is present in the block; storing the stream of image data in means for storing; initiating a selected operation with the printing system, in response to the presence of the token in the read block; and parsing the stream of image data to separate the stream into a plurality of image-related components.

When the stream of image data includes an image, the method may further comprise the step of separating the image from the stream of image data to form a second stream of image data. The method may then further comprise the step of storing the second stream of image data in a first storage section and the image in a second storage section. The method may further comprise determining that the image is stored at a location disposed remotely from the first and second storage sections and then may further comprise retrieving and processing the image stored at the remote location. The storing step may comprise updating the token to reflect that the bitmap is to be stored in the second storage section.

In a method in accordance with the invention, the image data source comprises a network across which the first stream of image data is communicated and said reading step is performed with a filtering arrangement, further comprising coupling the filtering arrangement with the network. The coupling step may comprise establishing multi-client parallel access between the filtering arrangement and the network so that a plurality of image data streams can be processed with the filtering arrangement.

When the printing system includes a printing-related device, a method in accordance with the invention may further comprise the step of setting up the device for a characteristic which serves to define the job. The method may further comprise the step of disposing the device remotely of the printing system. When the job characteristic is a font type, the method may further comprise the step of processing the font types. When the job characteristic is a finishing attribute, the method may further comprise the step of processing the finishing attribute.

By way of example only, embodiments of the invention willo be described with reference to the accompanying drawings, in which:

Figure 1 is a block diagram of a printing system;

Figure 2 is a block diagram of a processor/printer interface for the printing system shown in Figure 1;

Figure 3 is a block diagram of selected sections of a decomposer for the printing system of Figure 1, the selected sections including an input section, a system managing section and a parsing section;

Figure 4 is a schematic view of a prediction break table;

Figure 5 is a block diagram illustrating the manner in which bitmaps are stored in memory;

Figures 6 and 7 conjunctively represent a flow diagram depicting a preferred mode of operation for the input section, system managing section and parsing section of Figure 3;

Figure 8 is a flow diagram demonstrating the operation of a boundary code catcher ("BCC"), the BCC being used, preferably, in conjunction with either the input section or the parser;

Figure 9 is a block diagram illustrating the manner in which stored bitmaps are printed with the printer/processor interface of Figure 2;

Figure 10 is a block diagram of selected sections of a decomposer for the printing system of Figure 1, the selected sections including an input section with a decomposer image filter ("DIF"), a stream utility and the BCC, a system managing section and a parsing section;

Figure 11-12 represent, conjunctively, a flow diagram illustrating the manner in which DIF examines and processes a stream of image data communicated from a network to the input section; and

Figures 13A-13C are respective schematic views of an unmodified PDL stream, a modified PDL stream, and a bitmap.

Turning now to the drawings, and at this point especially to Figure 1, there is an electronic printing system 21 to illustrate a typical environment for this invention. In keeping with standard practices, the printing system 21 comprises a digital processor 22 having a main memory 23 and a mass memory 24, an input section 25 for providing a job written in a printer page description language (PDL), and a printer 26 for printing hardcopy renderings of selected image components obtained from the PDL. Furthermore, there is a user interface 27 for enabling a user to interact with the processor 22, the input scanner 25, and the printer 26.

As will be understood, the user interface 27 collectively represents the input devices through which the user enters image editing and manipulation instructions for the processor 22. Additionally, the interface 27 represents the output devices through which the user receives feedback with respect to the actions that are taken in response to the instructions that are entered by the user or otherwise, such as under program control. For example, the user interface 27 generally includes a keyboard or the like for entering use instructions, a monitor for giving the user a view of the process that is being performed by the processor 22, and a cursor

controller for enabling the user to move a cursor for making selections from and/or for entering data into a process that is being displayed by the monitor (none of these conventional components is shown).

The illustrated printing system 21 is centralized, so it has been simplified by assuming that all control instructions and all image editing and manipulation instructions are executed by the processor 22 under program control. In practice, however, the execution of these instructions may be handled by several different processors, some or all of which may have their own main memory and even their own mass memory. Likewise, either or both of the input scanner 25 and the printer 26 may have its own user interface, as indicated by the dashed lines 28 and 29, respectively. Indeed, it will be evident that the printing system 21 could be reconfigured to have a distributed architecture to operate with a remote input section and/or a remote printer (not shown). Data could be transferred from and to such remote input section and printer terminals via dedicated communication links or switched communication networks (also not shown).

As shown in Figure 2, the processor 22 preferably includes a PDL driver 31 for transferring to the printer 26 PDL descriptions of the electronic document files that are selected for printing. Thus, the printer 26 is illustrated as having a PDL decomposer 32 for decomposing such PDL descriptions to produce corresponding bitmapped image file. It will be appreciated, particularly in view of the discussion below, that the decomposer 32 is capable of receiving PDL files from mass memory, such as disk, or from off the network "on the fly." Additionally, the printer 26 includes a print engine 36, the print engine 36 including one or more image data buffering devices and being coupled with the decomposer 36 by way of an arrangement of imaging channels 34. The significance of the imaging channels will be discussed in further detail below.

Referring to Figure 3, an arrangement including the mass memory 24, the input section 25 and a portion of the decomposer 32 is shown. In the illustrated arrangement, the input section 25 comprises up to two PDL emitters, such as a workstation 40, or any other suitable PDL source 42. In one Example the workstation 40 is Xerox® 6085 (the term "Xerox 6085" is a trademark used by Xerox® Corp.) workstation coupled with a network 44, such as a network sold by Xerox® Corp. under the trademark of EtherNet™. The network is interfaced with the decomposer 32 by way of a suitable network interface 46, which could include one of many known interfaces such as TCP/IP™, AppleTalk™ or Token Ring™. Both of the workstation 40 and the PDL source 42 are interfaced with the decomposer 32 by way of a print server 48, which print server 48 can be a suitable protocol corresponding with the specifications of the PDL Source 42 and/or the network interface 46. The print server 48 communicates with a system manager 50.

The system manager 50 comprises a system state controller (SSC) 52 of the type disclosed in US-A-5,170,340, the user interface (UI) 27, which, in one example, is of the type disclosed in U. 5. Patent No. 5,083,210, a database manager 54 and a database front end processor 57. Additionally, the database manager 54 preferably comprises a database job queue and a database coalescer table. The features of the system can be obtained through use of any suitable, commercially available database. Alternatively, one of ordinary skill in the art could, without undue experimentation, construct the database of the system by reference to one of several known texts, such as the following text:

Martin, J.
Computer Data-Base Organization
Prentice Hall, Inc.
Englewood Cliffs, New Jersey
1975

The database front end processor 57, which serves to define the structure of the PDL job, and the sequence in which image related identifiers stored in the database are operated on, can be constructed by those skilled in the art in view of the type of database manager selected for use. As will appreciated from the discussion below, the database manager 54 is the conduit through which virtually all image related identifiers and all job identifiers ("handles") flow. Moreover, editing of postparsed information is achieved by use of a "make ready" process 58 which is coupled with the front end 57. The "make ready" process, which serves to interpret operator commands for performing bitmap processing, is used in the DocuTech™ electronic printer which is sold by Xerox® Corp.

Referring still to Figure 3, the print server 48, the SSC 52 and the data base front end processor 57 each communicate with a parsing section 60 by way of a preparse scheduler 62. Preferably, the steps of the parsing section 60 are performed on a MESA™ processor of the type manufactured by Xerox® Corp. The preparse scheduler 62 communicates with a preparser 64 by way of a block of shared memory 66. The preparser includes a manager section 68 and a main section 70. In one example, the manager section comprises a MESA™ processor of the type referred to immediately above, and the main section comprises a similar processor and a math coprocessor, the math coprocessor being similar to any one of a number of commercially available math coprocessors. Additionally, the manager 68 communicates with the print server 48 by way of a block of memory 71, and the main section 70 is configured to store a coalesce table, the significance of which table will be dis-

cussed in further detail below.

The preparser 64, which serves to break up PDL into image related components, is capable of storing those components in mass memory 24 which, preferably, is a disk storage device, such as the one used on the DocuTech™ electronic printer manufactured by Xerox® Corp. Preferably, the disk storage device is adapted to receive rasters or bitmaps in raster storage section 72 and internal PDL fragments ("internal PDL frags") in internal PDL frags. section 74. It should be recognized that internal PDL fragments are higher level primitives to be imaged on a substrate. In one example, an internal PDL fragment serves to transform coordinate systems inputted to the decomposer for printing. Employment of other storage sections in the mass memory 24 to receive other image related components is possible.

In practice, the mass memory 24 is configured to not only receive and store the image related components developed by the preparser 64, but to store, on a long term basis, a relatively large variety of logos (in logo DB storage section 76) and fonts (in a font storage section which is not shown). Employment of other storage sections in the mass memory 24 to store other image related components, on a long term basis, is possible. Additionally, the disk storage device preferably includes section 82 for storing PDL files to be processed and a section 84 for storing all information communicated to the database manager 54. Preferably, the storage section 82 is used in conjunction with volatile memory, such as cache so that each PDL job need not necessarily be stored out to disk. In one example, the job database 84 is adapted to contain a structure for image related components of a job written in a PDL. It will be appreciated by those skilled in the art that a suitable memory arrangement could be used in place of the job database without affecting the underlying concept.

In practice, the preparser 64 is interfaced with the raster section 72 by way of an image install process 86, a Boundary Code Catcher ("BCC") 87 and a raster directory service (DS) 88. Preferably, the BCC achieves its intended purpose through use of a plurality of programmable logic arrays programmed with suitable software, the details of the software being discussed in further detail below. Each raster or bitmap is stored in the storage section 72 with a corresponding break entry table ("BET") 73, an example of which break entry table is shown in Figure 4 as an array of break table segments. A detailed discussion of the format used for table 73 is provided in the following:

Title: *Xerox Raster Encoding Standard("*Encoding Standard")
Publication No. XNSS 178905
Publication Date: 1990

As explained in further detail below, each bitmap comprises an image, defined by one or more blocks of image data. Each block is divided into a plurality of segments with break entries, each entry designating a scanline count. Preferably, the BCC 87, in accordance with the the algorithm discussed below, builds the table 73 by indicating the location of each break entry in the image and relating it with a pointer to corresponding line control code.

The preparser 64 is interfaced with the logo DB section 76 (Figure 3) by way of a file finder process 90 and the internal PDL frags section by way of Internal PDL Frags Manager 92. Referring to Figures 3 and 5, one or more supplemented bitmaps are transmitted from the BCC 87 to the raster DS 88. Each supplemented bitmap comprises a bitmap with its corresponding BET. As illustrated specifically in Figure 5, each supplemented bitmap is assigned an image identifier and the image identifier is stored in the raster DS 88. Moreover, each image identifier points to one of the supplemented bitmaps stored in the storage section 72. Finally, for the illustrated arrangement of Figure 3, copies of the corresponding image identifiers from the raster DS 88 are passed to the preparser 64.

Referring still to Figure 3, the internal PDL frags manager 92 is adapted to assign an image identifier to each internal PDL fragment transmitted thereto, and pass each of those identifiers to the preparser 64. On the other hand, the file finder 90 assigns an identifier to calls for logos received at the preparser 64 from the print server 48, and permits the logos to be fetched by the preparser 64 when they are available in the logo DB section 76. When the called for logo is not available in the logo DB section, the file finder 90 is capable of issuing an appropriate fault message for display at the U1 27 or for printing with a hardcopy print.

Referring to Figures 6 and 7, the inputting and parsing of the PDL file(s) will be discussed in further detail. Referring specifically to Figure 6, at step 100, a job file written in a particular PDL, such as Interpress used by Xerox® Corp.,is provided from either the workstation 40 or the PDL source 42. Upon inputting the PDL job to the print server 48, basic information regarding, for example, the structure of the job and the order in which the job should be processed, is transmitted to the database manager 54 (step 102) by way of the preparse scheduler 62 and the database front end processor 57. The database manager 54 indicates to the SSC 52 that it has a job, and in due course, provided that the decomposer is ready (see step 104), the SSC 52 commands the print server 48 to begin transmitting blocks of the PDL file to the manager section 68 across shared memory block 71 (step 106) and causes the preparse scheduler 62 to obtain a job identifier ("handle") from the database manager 54 (step 108). The handle represents the information that the preparse scheduler 62

will need to pass the image related identifiers resulting from the parsing process on to the database manager 54. Alternatively, as illustrated by steps 104 and 110, if a job is ready to be inputted concurrent with a job being processed, the job that is ready to be inputted can be stored for subsequent processing. Preferably, as described in further detail below, step 110 is performed in conjunction with a filtering technique.

As the PDL is transmitted to the manager section 68 (step 112), it is, per step 114, broken down into global information, such as a header and a preamble, and page level information. Additionally, the manager 68 finds the beginning of each page within the job file ("master") for setting up the image related components ("data structures") to be received by the main section 70 . Essentially, the manager functions as a syntactical analyzer, insuring that the syntax of the encoded PDL master is correct. The manager preferably performs some pre-work for the main section 70 and possesses limited interpretative capabilities.

At step 118, page-level information is passed from the manager 68 to the main section 70 at which information, or, more specifically, data structures are created. The main section looks into a linked list of set up pages and sequentially decomposes them. The manager and the main section can work on separate pages or the main section can work ahead of the manager within a page. Once the manager has created a data structure for a page, the main section executes data structures within the page for storage in the mass memory 24. Per step 120, the main section 70 inserts any font names that it gleans from pages in a coalesce table stored in the main.

Referring to Figure 7, once the page-level data structures are available, identifiers therefore are obtained. If rasters (i.e., bitmaps) or references to rasters ("tokens of interest") are found among the data structures (step 122), the rasters are communicated to the raster. storage section 72 by way of the image install 86 and the raster DS 88. For each raster found among the data structures, a break entry table is created, with the BCC 87, at step 124. An approach for processing the rasters referenced by the tokens of interest is described in further detail below. Per step 126, the rasters, with their respective break entry tables, are assigned identifiers by the raster directory service, and the identifiers are passed to the main 70 for placement in the coalesce table.

If logo calls, i.e., calls for a merge item, are found among the data structures (step 128), the file finder 90 checks to see if the logos are in the logo DB section 76 (step 130). For those logos in the logo DB, corresponding pointers for the available logos are communicated to the main (step 132) for placement in the coalesce table. For those logos not in the logo DB, a fault message is, per step 134, returned to the main for eventual display or printing into hardcopy. Per steps 136 and 140, internal PDL fragments are created and communicated to the internal PDL frags storage section 74 by way of the internal PDL frags manager 92. With step 140, the internal PDL fragments are assigned identifiers or "tokens" by the internal PDL frags manager 92, and the identifiers are passed to the main section 70. As shown by steps 142 and 144 any other data structures flowing from the main section 70 would be handled in a manner similar to that shown for rasters or internal PDL fragments. After all of the identifiers are provided to the main section 70, a signal is sent to the preparse scheduler 62, and the identifiers, per step 146, are communicated to the database manager 54.

Referring to Figure 8, a technique for creating the break entry table 73 of Figure 4 is discussed in further detail. One or more data blocks, representing a bitmap, are buffered and, initially, at step 200, the BCC 87 fetches a data block, along with pertinent parameters, such as "image handle", "pixel sequence type", "pixel encoding offset", etc. The BCC 87 then reads one word of image data at-a-time (step 202), from the block of image data, and stores any line boundary code. If a word is not line boundary code (step 204) then the process returns to step 202. If, on the other hand, the word is line boundary code, then the word is examined, per step 206, and it is determined whether the word is at the end of a segment. If the word marks the end (or, alternatively, the beginning) of a segment, then a break table entry is generated (step 208) and a check is performed, at step 210, to determine if the entire image of the bitmap has been examined. When the end of the image is reached (step 212), the current break table entry is designated as the "Last Break Table Entry". If the word does not mark the end of a segment (step 206), but the word is at the end of the block being examined (step 214), then the process returns to step 200 to fetch another block of image data, provided another block is in the image install 86. When the word is neither the end of a segment nor the end of a block (steps 206, 214), a check is performed to determine if the end of the image has been reached. When the end of the image has been reached, the process ends, as described above, otherwise the process returns to step 202.

As described in EP-A-0,574,224, a postparser (not shown) causes all of the data structures assimilated by the decomposer to be placed in a suitable form for printing and places corresponding raster identifiers, font identifiers and primitive representations of internal PDL fragments into a bandlist for printing. Through employment of the bandlist, image data is delivered to the imaging channel arrangement 34, each of which channels is adapted to process, and, more particularly, decompress segments having up to 256 scanlines of image data.

Referring to Figure 9, the break entry table is read by the imaging channel arrangement 34 and a suitable

number of segments are delivered, in parallel, to the imaging channels for decompression of the compressed scanlines of each segment. In one example, each segment comprises 16 scanlines of image data. As the image data is decompressed, the resulting decompressed image data can be buffered for subsequent consumption by the print engine 36.

It should be recognized that the decompressed image data can be used for purposes other than printing. For example, the decompressed image data could be simply displayed on the UI 27. Additionally, since segments of each bitmap are decompressed in parallel, portions of a bitmap can be displayed, or printed, out of sequence. This sort of selective output can be particularly useful in, among other operations a cut and paste routine. More particularly, certain editing functions can be performed with selected portions of the bitmap, rather than the entire bitmap itself.

Referring to Figure 10, another arrangement of the input area for the printing system 21 is shown. The illustrated arrangement of Figure 10 is similar to that of Figure 3 except that an input filtering arrangement 220 is substituted for the shared memory 71. The input filtering arrangement 220 comprises a decomposer input filter ("DIF") 222 communicating with both a stream utility 224 and a BCC 87'. In practice, the BCC 87 and BCC 87' are structurally and functionally equivalent; however, the two apparatuses are designated with separate numerals for ease of discussion and purposes of clarity.

The functions of DIF 222 and the stream utility 224 can be obtained on a MESA processor of the type designated above. Moreover, it will be appreciated by those skilled in the art that the functions of DIF and the stream utility could be combined.

Referring still to Figure 10, the operation of the filtering arrangement 220 is explained in further detail. It should be recognized that while the following description refers to the processing of a single stream of image data, the input section 25, along with the filtering arrangement 220, are, as mentioned immediately above, capable of processing multiple streams, in parallel. Upon receiving a stream of image data, the network interface 46 indicates, to the print server 48 that a connection is desired. In response to such indication, the print server passes suitable parameters to the DIF 222 for processing the incoming input stream. In turn, DIF passes appropriate "handles" to the stream utility 224 for accessing the input stream and passing the by-products of the filtering process to either the raster storage section 72 or the PDL storage section 82. Upon setting up the DIF 222 and the stream utility 224 appropriately, the input stream is inputted to DIF via the network interface 46 through employment of the stream utility 224.

Referring to Figures 11 and 12, the processing of the input stream by DIF 222 is discussed in further detail. Preferably, image data is read by DIF (step 226), block-by-block, as it is fed thereto with the stream utility 224 It will be recognized by those skilled in the art that a block could be as small as a single word, or that more than one block could be fed to the DIF at one time. As each block is read, DIF determines, via steps 227, 228, whether the block under examination includes a token of interest. For each new block, the step 227 initially causes the first token found in the new block to be referenced. At step 229, the process checks to see if the end of the block has been reached. It has been found that a counting index scheme can be used to determine the beginning and end of the block. If the end of the block has not been reached, then the processs loops back to step 227 for getting the next token. If, on the other hand, the end of the block has been reached, then, via step 230, that block is handed over to the stream utility 224 for transmission thereby to PDL storage 82, and the process loops back to step 226 for reading another block. As indicated above, the PDL storage section 82 can include cache. Accordingly, the block need not necessarily be stored out to disk.

Once a token of interest is found (step 228), it is determined, at step 232, whether the found token constitutes the end of the input stream. If the token does not constitute the end of the stream, then it is determined, at step 234, whether the token relates to a reference to image data outside of the stream ("outside data"), such as a sequence insert file ("SIF") or a sequence insert master ("SIM"). A detailed discussion of SIFs and SIMs can be found in the following:

Harrington, S. J. and Buckley, R. R.
Interpress: The Source Book
Simon & Schuster, Inc.
New York, N. Y.
1988

Assuming that the token of interest designates outside data, such outside data is, via step 236, retrieved and processed suitably. In one example, the outside image data would comprise a bitmap stored either locally or remotely of the printing system 21. The referenced bitmap would preferably be processed in accordance with steps 242, 244, 246, 248 and 250 (Figure 12), as described below. It will be appreciated by those skilled in the art that step 236 contemplates the retrieval and processing of other local and/or remote images other than bitmaps. For example, the outside data could constitute another stream of image data.

When it is determined that the token of interest corresponds to a bitmap (step 240) (Figure 12), the process

proceeds to step 242. If the bitmap is smaller than a predetermined size, then the bitmap is left in the stream for parsing and eventual storage with the image install 86. It has been found that storing bitmaps with the filtering arrangement 220 does not necessarily facilitate the decomposing process unless the bitmap to be separated is above a predetermined threshold size. If the bitmap is below the predetermined threshold, then it is eventually stored at step 230. When the bitmap is greater than the predetermined threshold then, it is examined with the BCC 87' (step 244) in accordance with the procedure described above. Additionally, DIF 222 provides the image data associated with the bitmap to the BCC 87'. Once a break table is formed for the bitmap, the bitmap is stored, along with the break table, by the stream utility 224 (step 246) in the raster storage section 72. Upon storing the bitmap and break table, the token of interest is revised or updated (step 248) to reflect a modification of the input stream. Storing the bitmap removes a plurality of blocks from the stream and these blocks need not be read again at step 226. Accordingly, through use of step 250, the examination is continued at a block which is at the end of the bitmap.

Referring to Figures 13A-13C, the manner in which the input or original PDL stream is modified with the filtering arrangement 220 is discussed in further detail. Referring specifically to Figure 13A, a stream fragment 254', including PDL data portions and image information, is shown. The image information, which, in one example, comprises pixel vector information (i.e., bitmap information), is designated with the numerals 256', 258' and 260'. The section 256', which is commonly referred to as a "token", indicates a pixel sequence type, i.e., whether the image is a pixel vector, a SIF/SIM, etc. While Figures 13A-13C show the pixel sequence type as being a bitmap, as mentioned above, the pixel sequence type could comprise one of a variety of image types. Additionally, as described in further detail below the token 256' can serve to initiate a large range of job characteristics. The section 258' preferably is one to three bytes in length and designates the length of the bitmap. It should be appreciated that, in another aspect of the disclosed embodiment, as described below, the section 258' could include a variety of information serving to further define a process indicated by the token. The information regarding length is employed to make an appropriate decision at step 242 of Figure 11. The portions designated with the numerals 260' represent portions of the stream which are not necessarily stored in the raster storage section 72, while the N blocks between the portions 260' represent portions of the stream which are stored in the raster storage section 72. It will be appreciated by those skilled in the art that the portions 260'could be stored with the N blocks, or deleted altogether.

Referring to Figure 13B, the input stream of Figure 13A, as modified in accordance with the present procedure, is designated by the numeral 261'. The image information of the modified stream 261' is designated with the numerals 260', 260", 262', 263' and 264'. A modified token, indicating a preinstalled pixel sequence type, is provided in section 262'. A length designator 263' serves as an offset indicator to the next valid PDL token. Preferably, an eight byte file identifier, being designated with the numeral 264' and pointing to the location of the pixel vector in the raster storage section 72, is provided. In the modified stream, the file identifier may be written over part of portion 260' to form portion 260".

Referring to Figure 13C, a pixel vector file, as stored in raster storage section 72, is designated with the numeral 265'. Various portions of the stored pixel vector file 265' are respectively designated with the numerals 266', 268', 270' and 272'. Portions 266' represent extraneous material resulting from page boundarizing while portion 268' represents image data corresponding substantially with the stored pixel vector. Additionally, portion 270' represents a break table corresponding with the image data and portion 272' comprises a pixel vector trailer, the trailer preferably being written at the end of the pixel vector file. As indicated in Appendix C, the pixel vector preferably relates to a host of information required by the raster directory service 88 for installing the pixel vector in the raster storage section 72, such information including sequence length, pixel sequence type, compression scheme, etc.

Referring again to Figure 10, in one example of operation, a plurality of connections are made at the filtering arrangement 220, each stream is processed accordingly and a first stream is fed, with the stream utility 224, to the manager 68 for parsing. The other streams are modified, in accordance with the above-described procedure, and the stream utility 224 directs the modified streams and the separated bitmaps to the PDL storage 24 and raster storage section 72, respectively. As mentioned above, the PDL storage section can include cache memory so that one or more streams need not be stored out to disk. The size of the cache employed is only limited by practical constraints.

While parsing image data, if the manager 68 detects a bitmap identified by sections 262' (Figure 13B), it causes the identified bitmap in the raster storage section 72 to be registered in the Raster DS 88 by the image install 86. It should be appreciated that in the illustrated arrangement of Figure 10, the BCC 87 only forms break tables for those bitmaps which are less than a threshold size. For this arrangement, the BCC 87 is relegated to a minor role since a substantial number of the bitmaps fed to the image install 86 are already appended with respective break tables, each of these appended break tables being developed with the BCC 87'. Each bitmap of the first stream, with its corresponding break table, is installed in the raster storage section

72 as described above with respect to the discussions of Figures 8 and 13C. Once the first stream is parsed, each modified stream stored in the PDL storage section 82 is, in turn, retrieved with the stream utility 224 and parsed.

Referring to the following table, an example which serves to illustrate a feature of the filtering arrangement 220 is shown..

Event in msec	With Filtering Arrangement	With Spooling
Start Capture PDL	0	0
End Capture PDL	43339	39434
Start PreParse	43394	39481
End PreParse	45255	48531

The exemplary results demonstrate that capture time of the PDL input stream is shorter for the case in which the stream is spooled directly to the PDL storage section 82 than for the case in which the stream is filtered and spooled. On the other hand, the time required to preparse is greater for the case in which the stream is spooled directly to the PDL storage section 82 than for the case in which the stream is filtered and spooled. Accordingly, despite the additional time required to filter a stream, the time required to both capture and preparse the stream which is filtered and spooled can be less than the time required to both capture and preparse the stream which is spooled directly.

Referring again to Figure 13A, it should be appreciated that, in another aspect, the token 256' can serve to initiate an operation other than storing N blocks of image data in the storage section 72. In other words, in the most generalized aspect of the disclosed embodiment, a token and, optionally, correponding information can be read by the DIF 222 and used to initiate a selected operation in the printing system 21. More particularly, the token, with the optional corresponding information, can define the job, represented by the stream of image data, in that the token can serve to set up a device for a "job characteristic".

Referring again to Figures 11-12, the manner in which devices are set up for job characteristics is discussed in further detail. After it is determined that the token of interest is a job characteristic to be used in setting up a device (step 260), such as a font developing arrangement or a finishing arrangement, the process checks, at step 262, to determine if the job characteristic is a "font-related job characteristic".

Provided that the job characteristic is font-related, the stream utility 224 (Figure 10) initiates a font development procedure. In one example, a known font checking procedure is used to determine if a suitable bitmap is available in a font database, the font database being disposed locally or remotely of the printing system 21 (Figure 1). An arrangement for implementing such font checking is disclosed by EP-A-0,574,224, U. S. Patent No. 5,113,355 and/or U. S. Patent No. 5,167,013. The advantage to checking for fonts with the stream utility 224 is that the printing system can indicate to the operator that a font is not available prior to the time of coalescing the job. In another example, the stream utility 224 can initiate a font rendering routine, prior to coalescing. Examples of font rendering can be found in the above-referenced text entitled *interpress: The Source Book.*

If the job characteristic is not font-related, then the process determines, at step 266, whether the job characteristic is finishing-related, i.e. whether the job characteristic is a "finishing attribute". When the job characteristic is finishing-related, a finishing arrangement associated with the printer 26 (Figure 1), is set up for the selected finishing attribute. An exemplary finishing arrangement, suitable for use with the printer 26,can be found in U. 5. Patent No. 5,045,881. The finishing arrangement could be set up for the selected finishing attribute, with the stream utility 224 (Figure 10), in a mannner consistent with the scheme disclosed by U. S. Patent No. 5,129,639.

If the job is a job characteristic other than a font related job characteristic or a finishing-related job characteristic (step 270), then the other job characteristic is processed at step 272. Various other job characteristics could include PDL file library references, printing instructions, etc. PDL file library references would be used to fetch remote libraries for use in decomposing. As explained in the following, a library is appended to a PDL file to provide macro definitions for various operations:

Adobe Systems Incorporated
PostScript® Language Reference Manual
Addison-Wesley Co.
1990

Additionally, the token can be used to determine job disposition. For example, the token can provide an

indication regarding whether the job is going to a job file or a print queue. An exemplary job file and print queue are disclosed in U. S. Patent No. 5,164,842. Knowledge regarding where the job is to be directed allows for optimization of memory usage, such as optimization of cache usage.

Finally, the token can be used to initiate an operation with respect to a referenced file, the referenced file being disposed locally or remotely of the printing system 21. For example, the token can initiate the fetching of a sequence insert master.

In view of the above description, numerous features of the disclosed printing system will be appreciated by those skilled in the art.

One feature of the disclosed printing system is to minimize the amount of time required to parse one or more incoming PDL input streams. In particular, for a significant number of network jobs, overall parsing time is reduced as a result of filtering and preinstalling each bitmap in the raster storage section.

Another feature is storage of a plurality of jobs, based on modified streams, in a PDL storage section. Since the modified streams are devoid of bitmaps exceeding a selected threshold size, the time interval required to read the job from the PDL storage section is minimized. Moreover, by "stripping" a significant number of bitmaps from the input streams, the resulting bandwidth of the PDL storage section is minimized.

Yet another feature is to service a plurality of users, in parallel for enabling efficient processing of parallel input streams. For a network printing system managing many jobs in a relatively short time interval, such parallel processing improves overall performance of the system.

Yet another feature is to redistribute work throughout the printing system, i.e., the "pipeline" More particularly, the work is redistributed from a computationally burdened section, namely the system input area. This sort of redistribution is particularly useful in processing bitmap ladened jobs.

Yet another feature is to filter a wide variety of images contained in input streams. In particular, the disclosed printing system is capable of locating and processing images referenced in the input streams. In one example, such references can relate to sequence insert files and/or sequence insert masters, each of which can be local or remote. Additionally, the input stream can reference another input stream.

Yet another feature is to make various operations of the input section, system managing section and parsing section transparent to the DIF. That is the dual burdens of coordinating image data flow and managing control are taken away from DIF and initiated advantageously by the stream utility. Accordingly, the level of complexity residing in DIF is reduced significantly.

Another feature is to set up various devices associated with the printing system for one or more job characteristics. In one example token(s) can be used to set up a font development scheme,and/or a finishing arrangement. More particularly, when DIF reads a selected token, it initiates one or more devices, with the stream utility, for one or more of the job characteristics.

Claims

1. A printing system for producing prints from a job represented by a stream of image data written in a page description language and having a token expressed as a plurality of bits, the stream of image data being generated and transmitted from an image data source, comprisng:

   an input section, communicating with the image data source, for receiving the stream of image data, said input section including means for reading a block of image data to determine the presence of the token in the block of image data;

   means, communicating with said input section, for storing the stream of image data;

   means, responsive to said reading means determining the presence of the token, for initiating a selected operation in said printing system after said input section receives a portion of the stream of image data; and

   a parser, communicating with said storing means, for parsing the stream of image data to separate the stream into a plurality of image-related components.

2. A printing system as claimed in claim 1, wherein the parsing occurs at a time later than the time for performing the selected operation.

3. A printing system as claimed in claim 1 or claim 2, in which the token indicates that the stream of image data includes a bitmap, wherein said input section includes means for separating the bitmap from the first stream of image data for forming a second stream of image data.

4. A printing system as claimed in claim 3, wherein said storing means comprises;

a first storage section; and

a second storage section with the second stream of image data being stored in said first storage section and the bitmap being stored in said second storage section.

5. A printing system as claimed in claim 4, further comprising means, cooperating with said parsing means, for assigning a bitmap identifier to the stored bitmap.

6. A printing systm as claimed in claim 4 or claim 5, wherein the second stream includes information designating a storage location for the bitmap in said storing means.

7. A method of producing prints with a printing system from a job represented by a stream of page description language and having a token expressed as a plurality of bits, the stream of image data being generated with an image data source and transmitted from the image data source, comprising the steps of:

reading a block of image data to determine whether the token is present in the block;

storing the stream of image data in means for storing;

initiating a selected operation with the printing system, in response to the presence of the token in the read block; and

parsing the stream of image data to separate the stream into a plurality of image-related components.

8. A method as claimed in claim 7, wherein said parsing step is performed at a time later than the time for performing the selected operation.

9. A method as claimed in claim 7 or claim 8, in which the stream of image data includes an image, further comprising the step of separating the image from the stream of image data to form a second stream of image data.

10. A method as claimed in claim 9, further comprising the step of storing the second stream of image data in a first storage section and the image in a second storage section.

11. A method as claimed in claim 10, further comprising the step of assigning an image identifier to the stored image.

12. A method as claimed in claim 10 or claim 11, further comprising the step of determining that the image is a bitmap.

13. A method as claimed in claim 12, wherein said storing step comprises updating the token to reflect that the bitmap is to be stored in the second storage section.

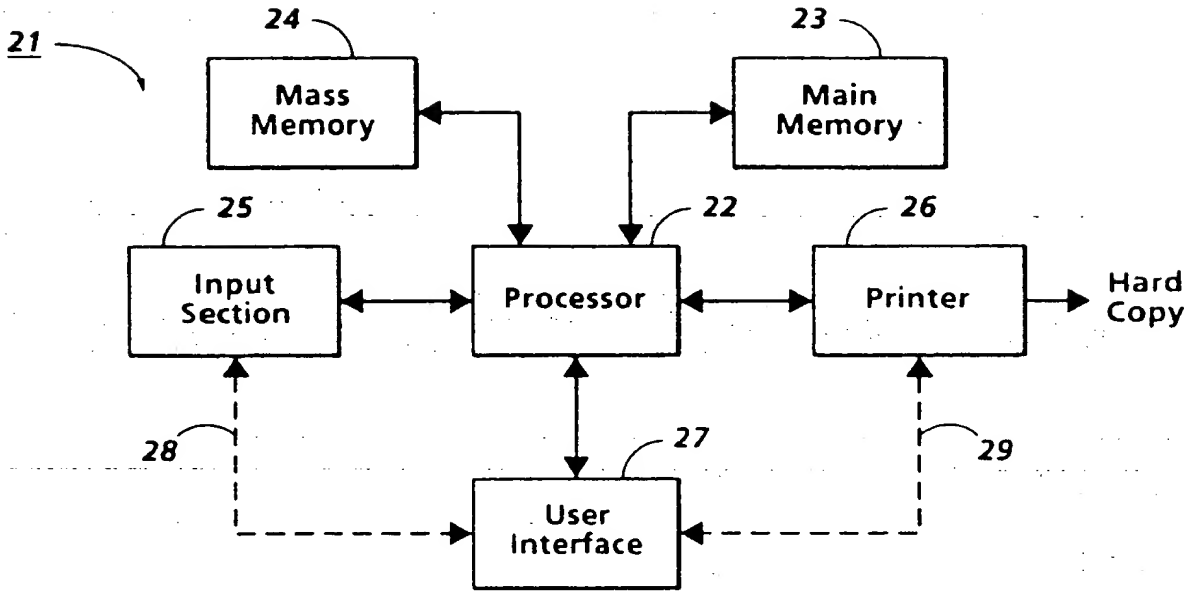


FIG. 1

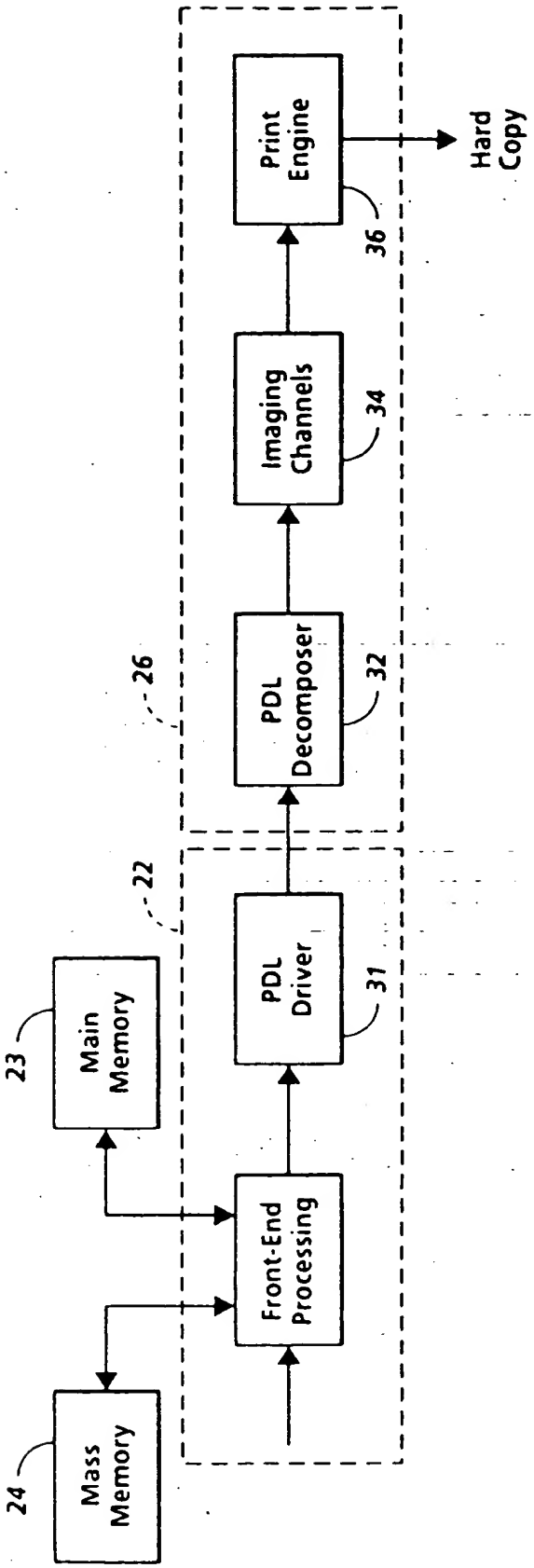


FIG. 2

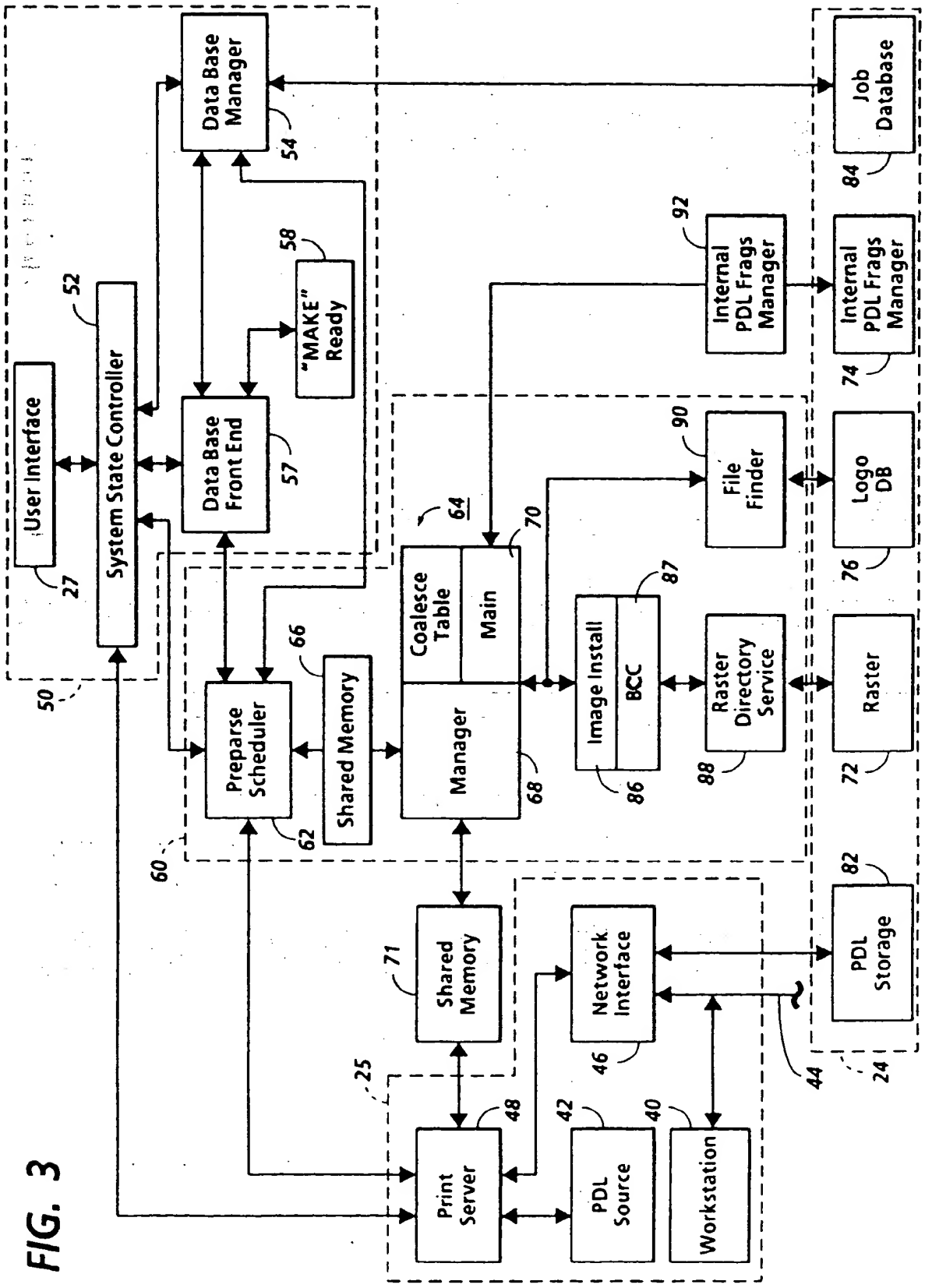
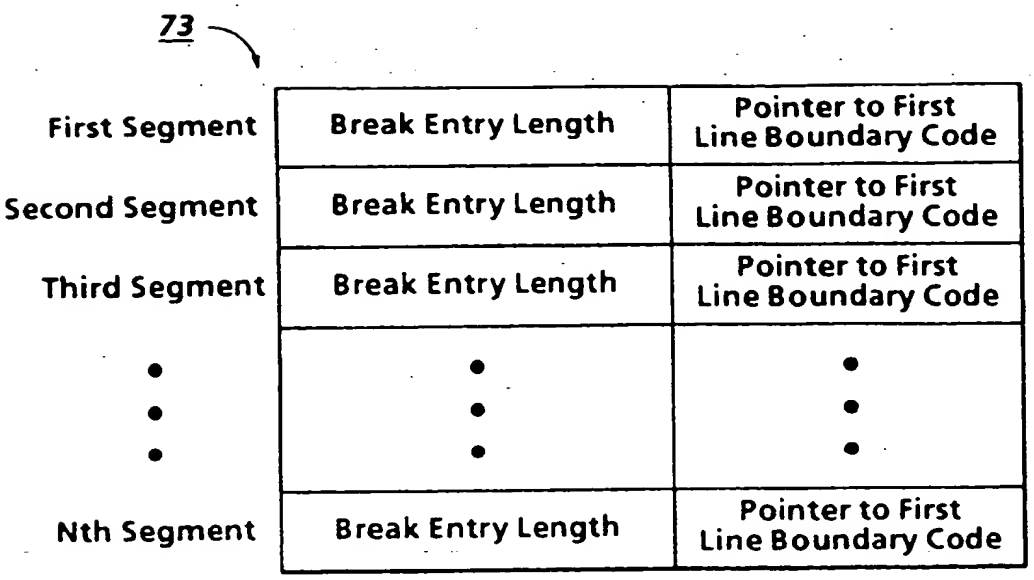


FIG. 3

FIG. 4



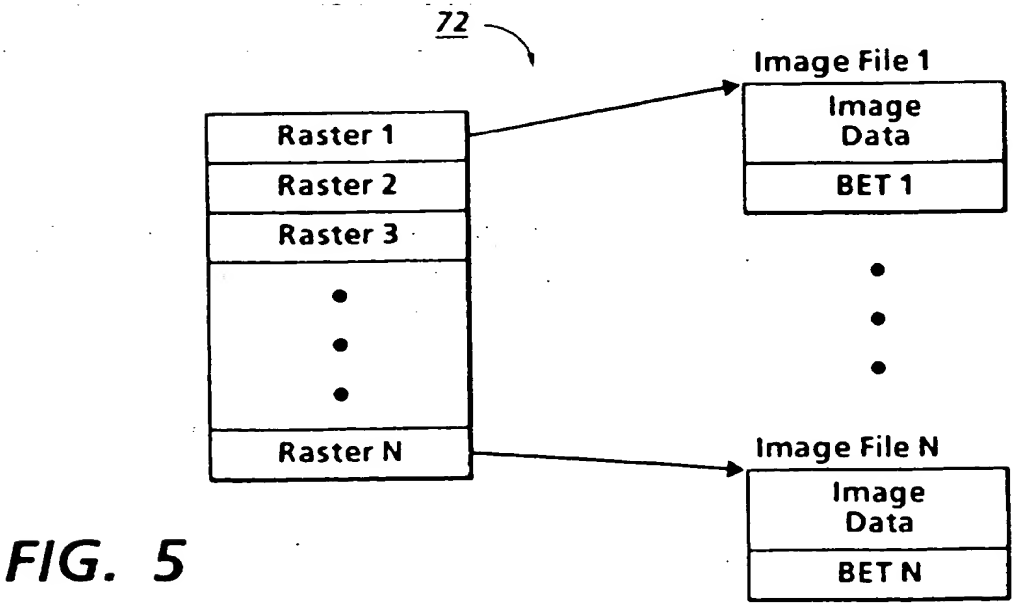


FIG. 5

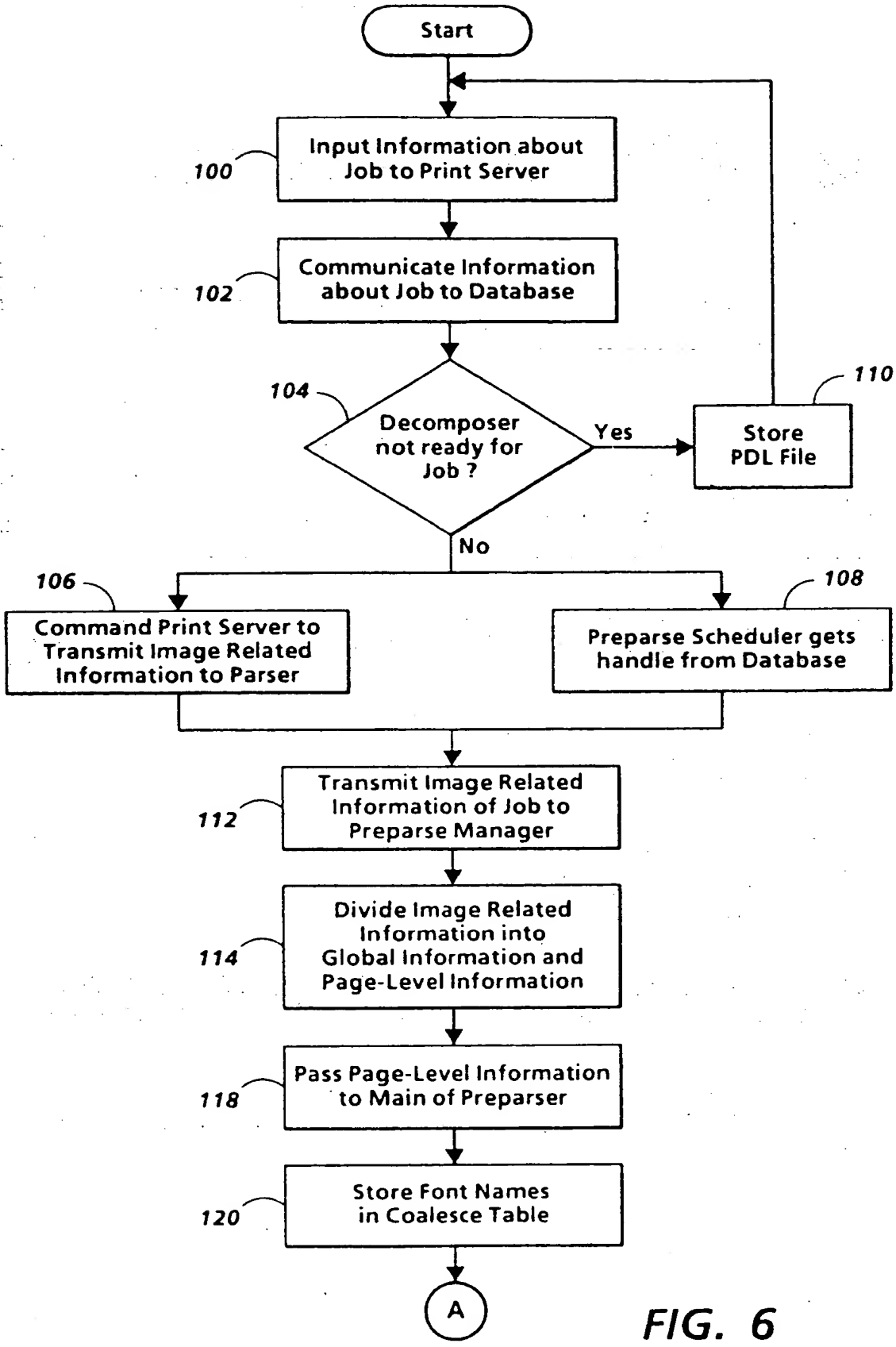


FIG. 6

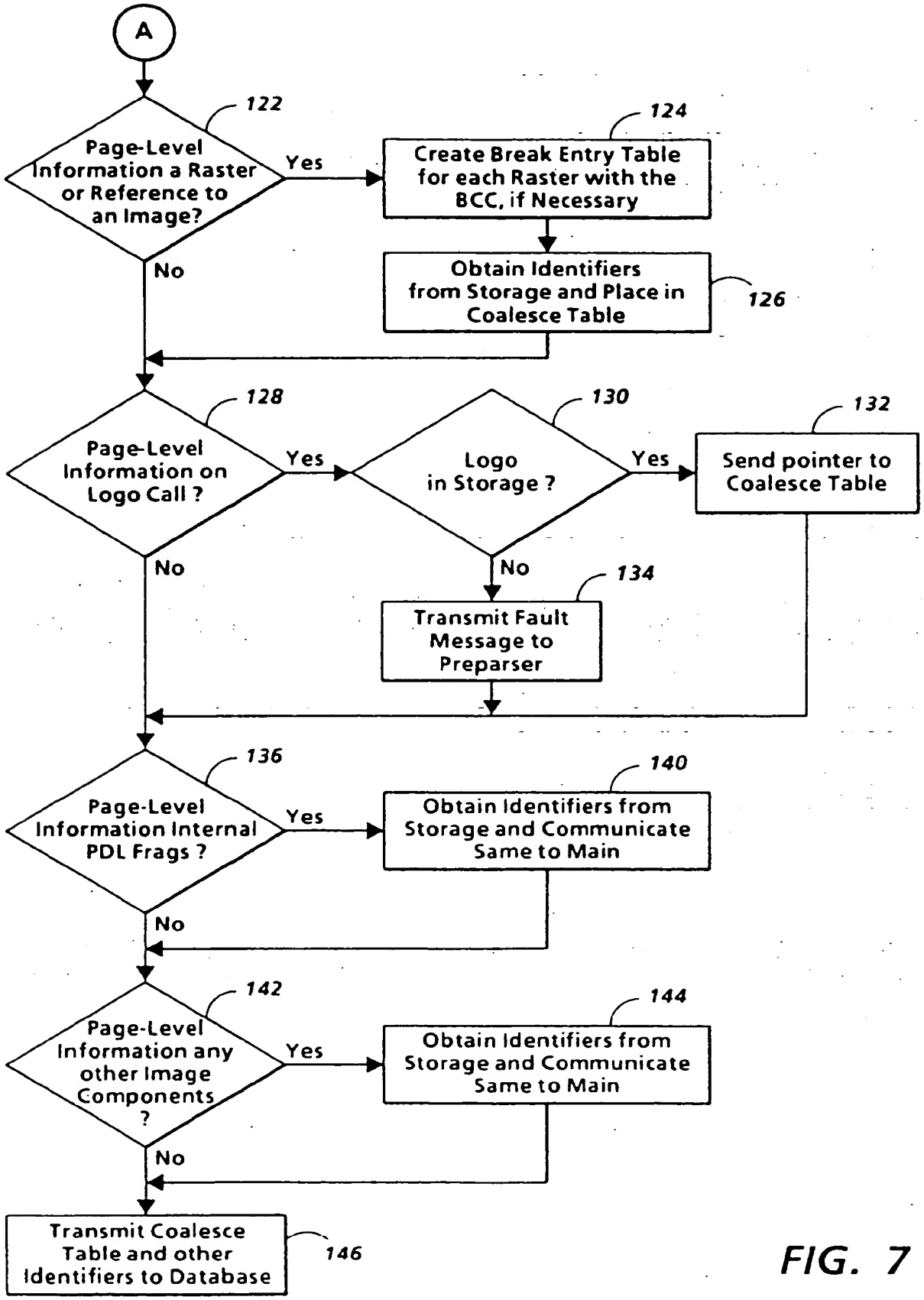


FIG. 7

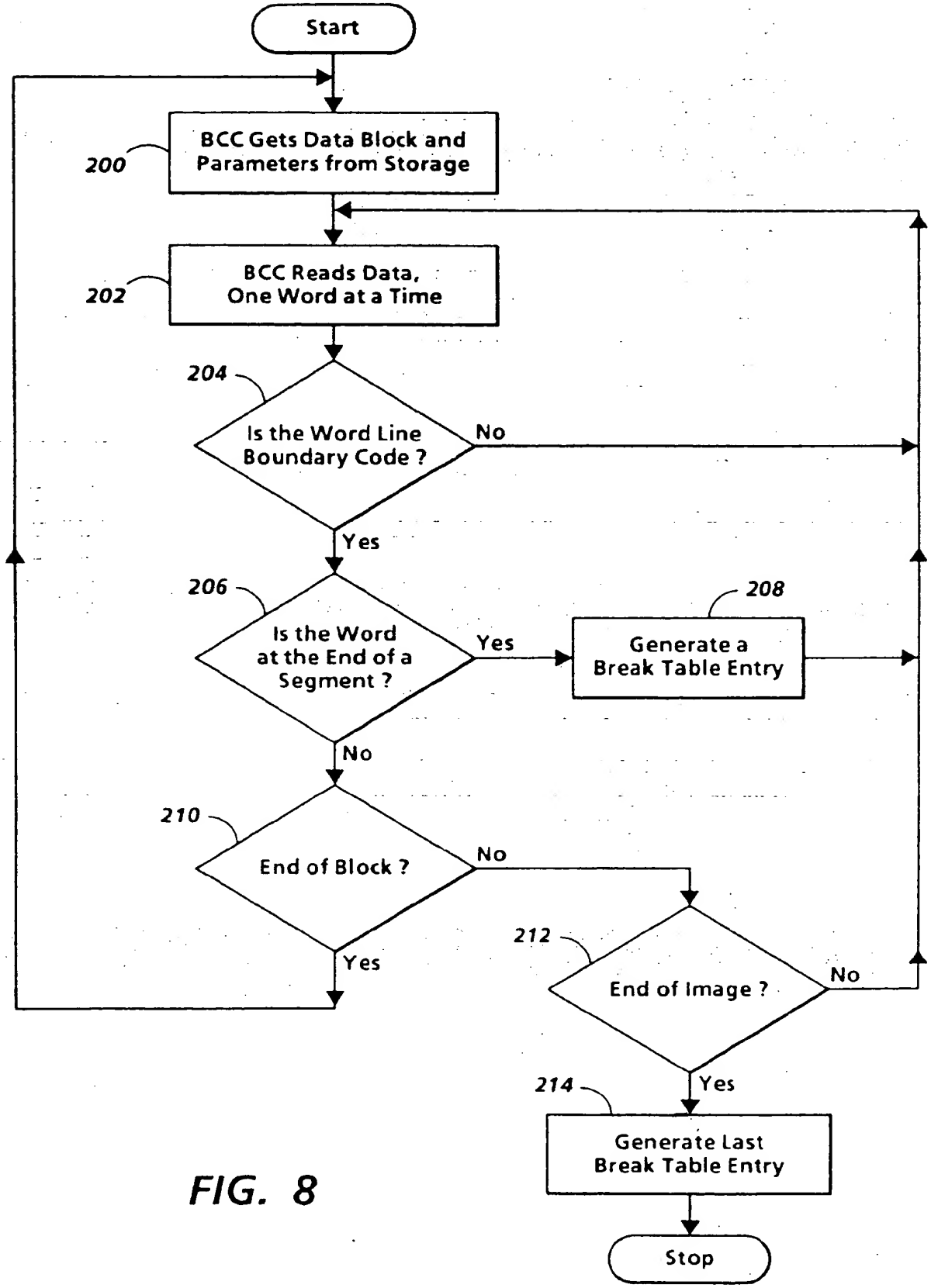
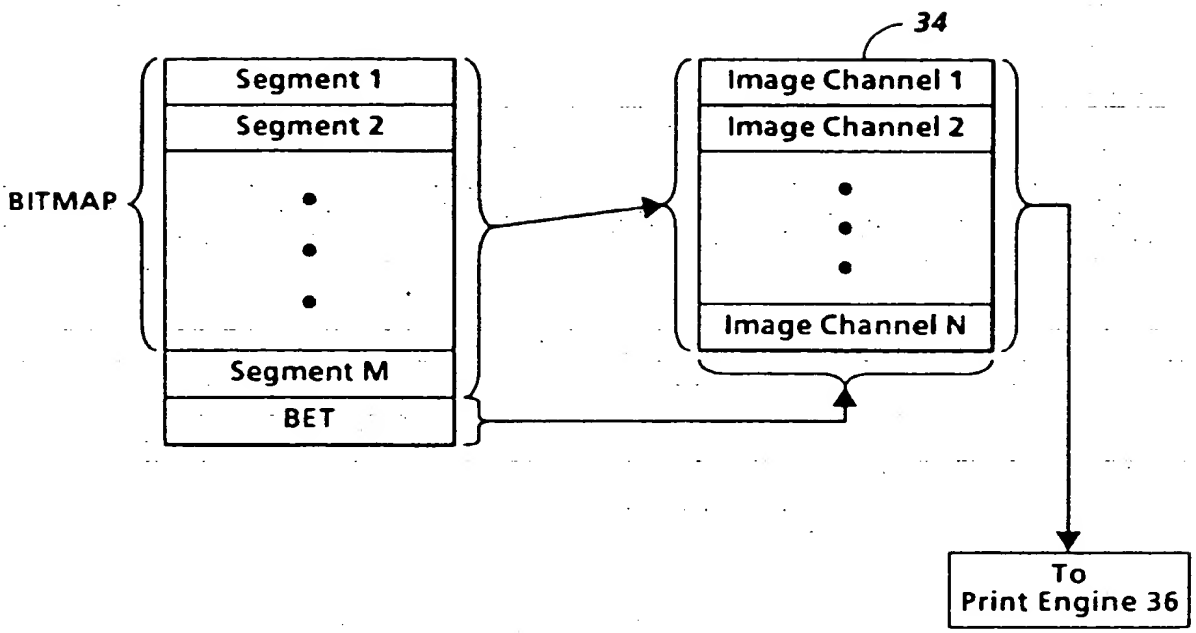


FIG. 8

FIG. 9



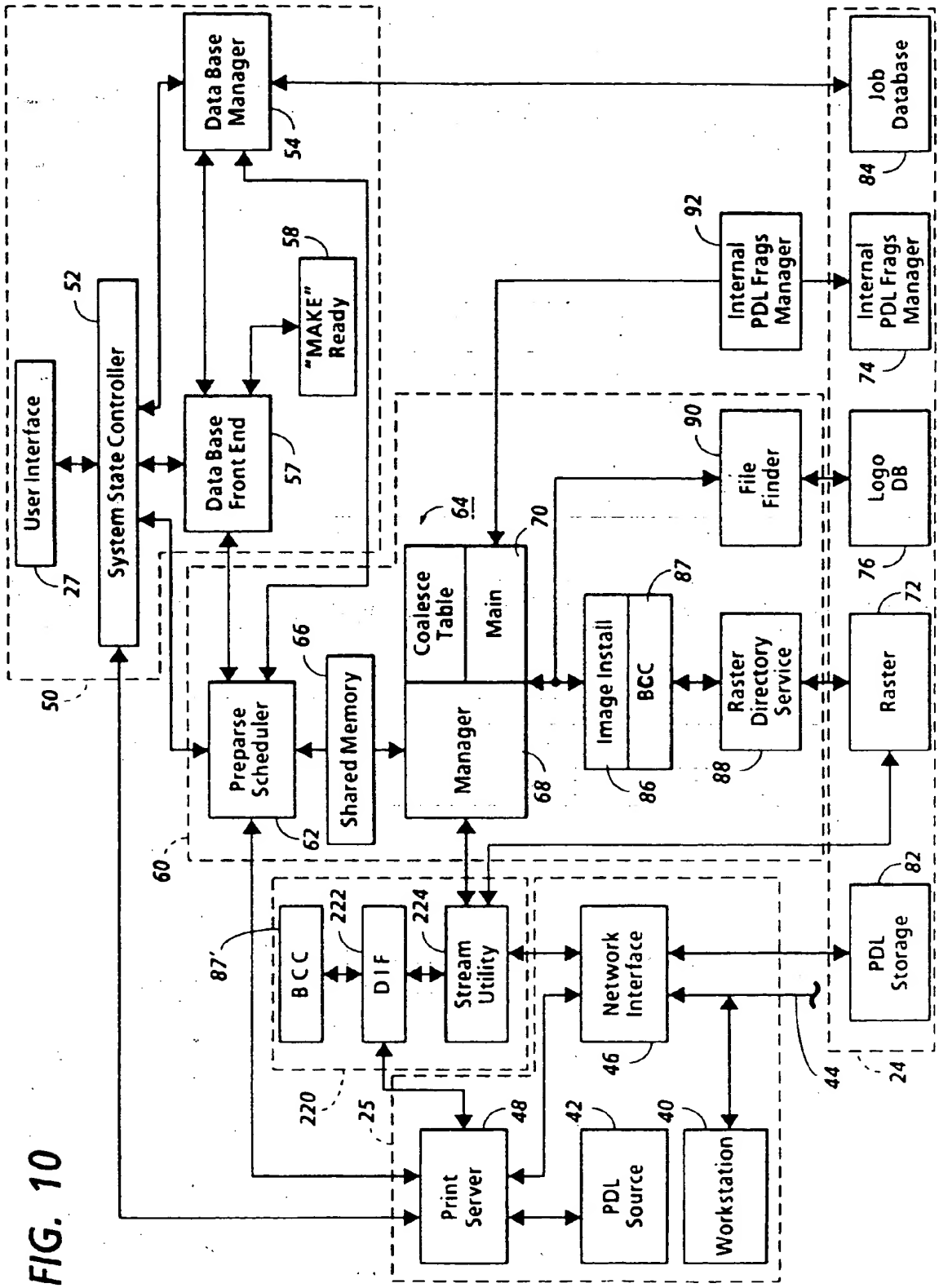


FIG. 10

FIG. 11

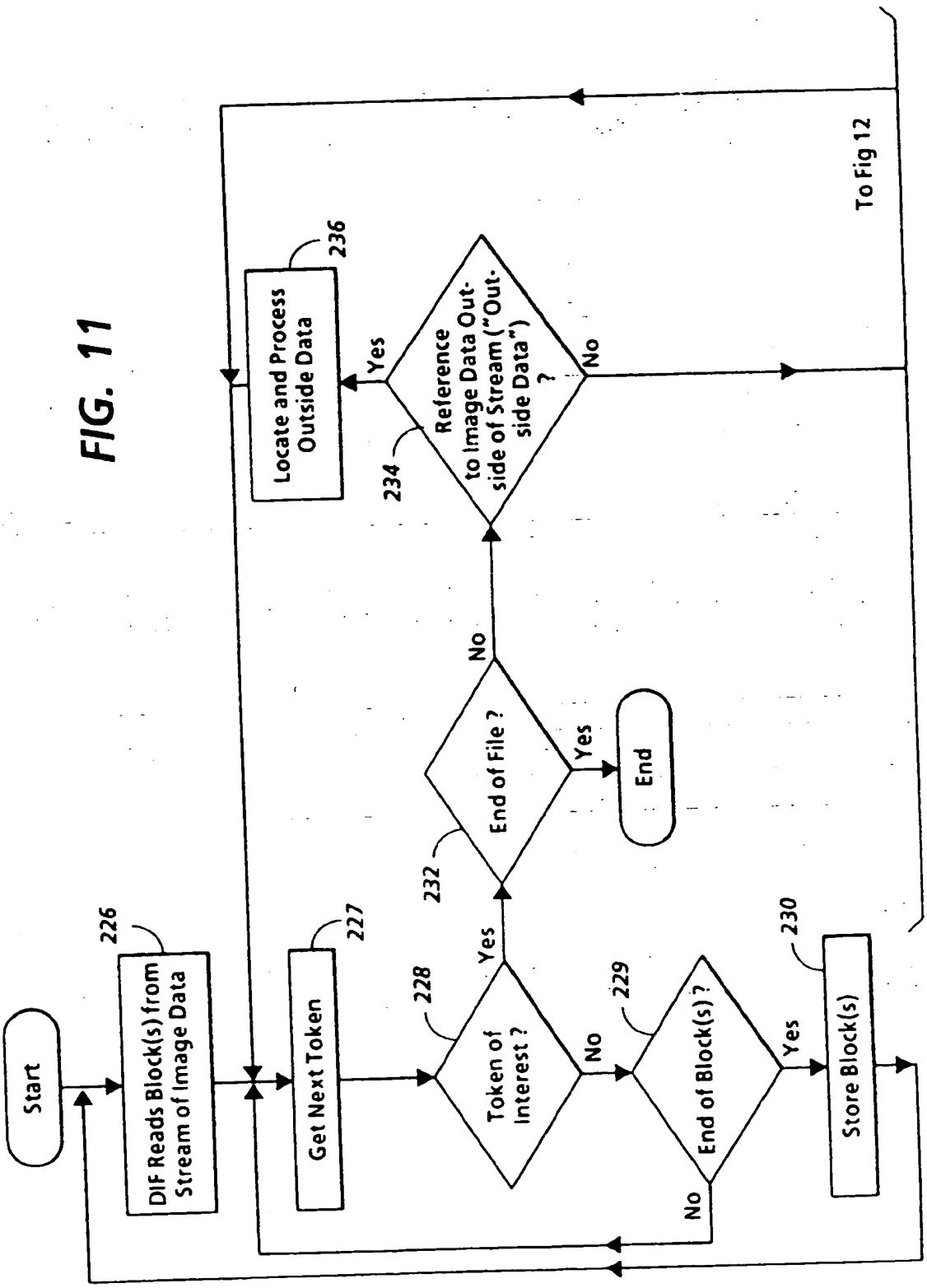
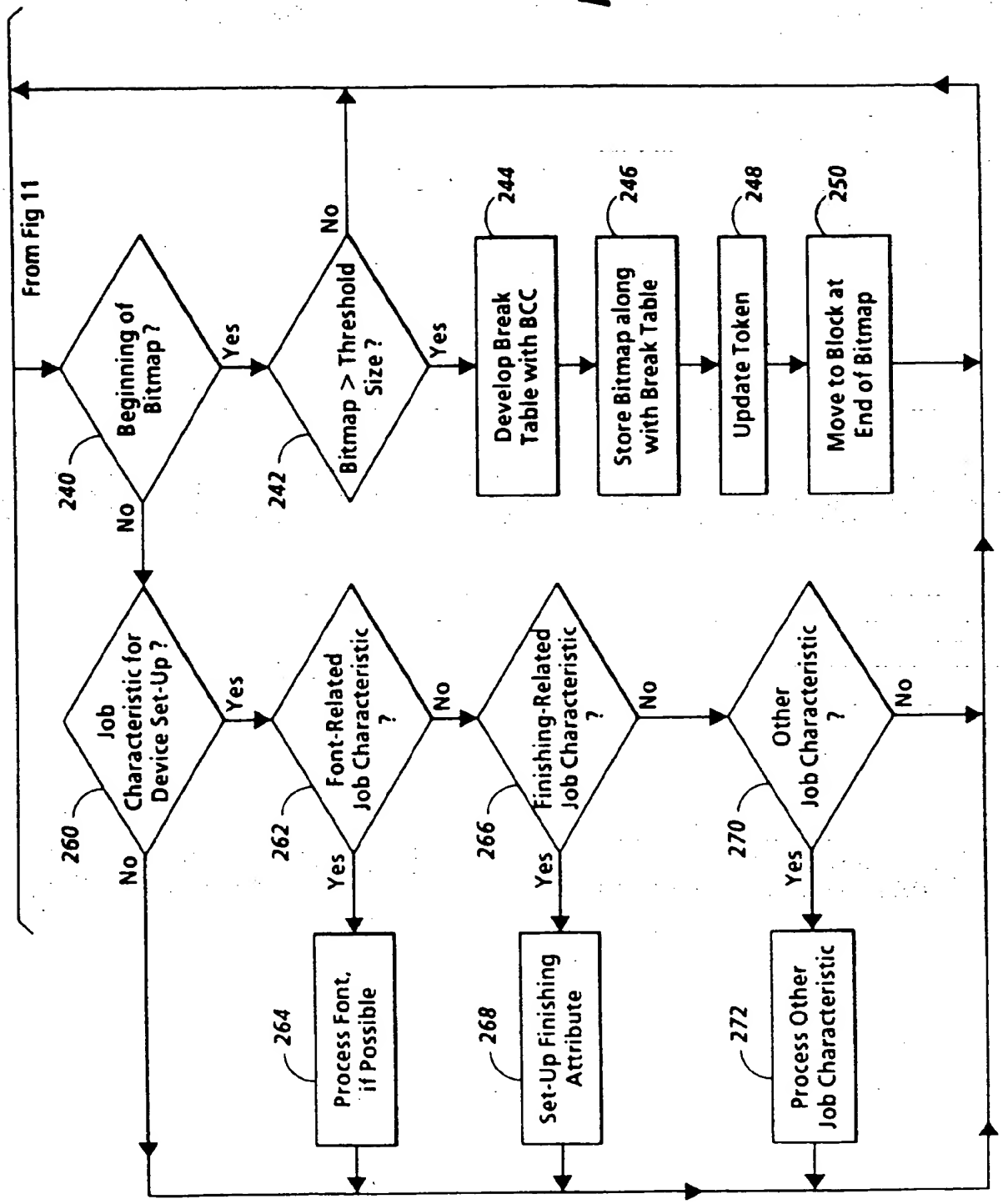


FIG. 12



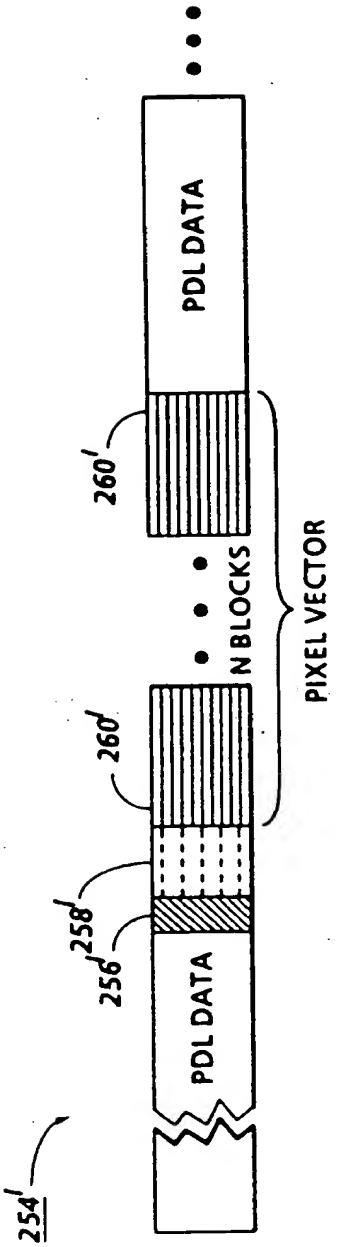


FIG. 13A

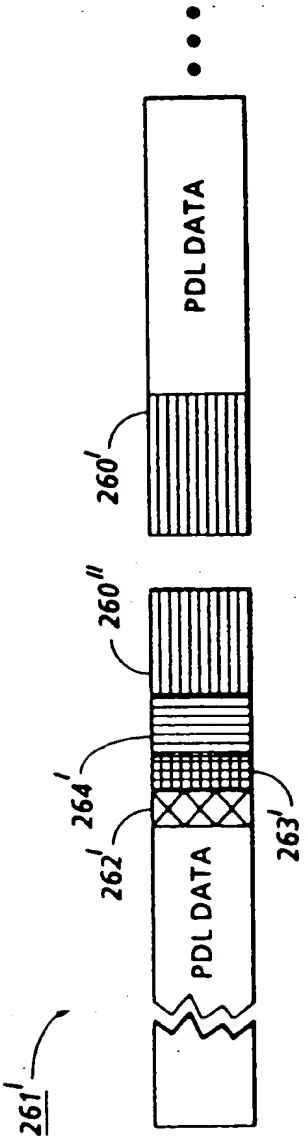


FIG. 13B

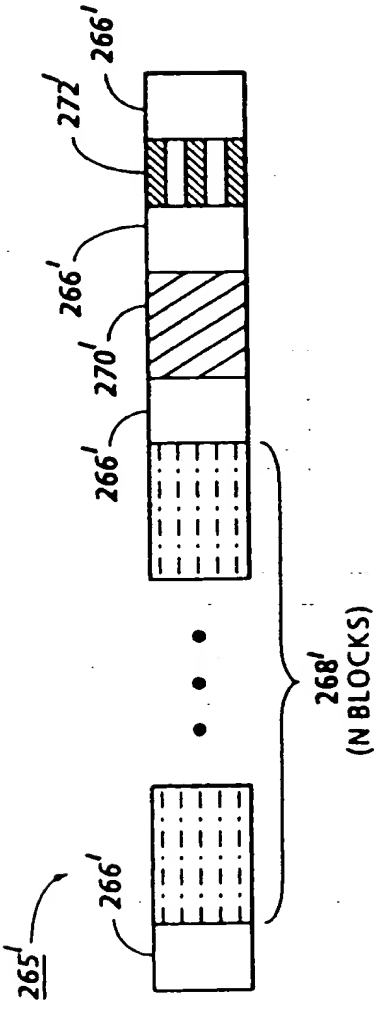


FIG. 13C